# Intelligent Systems Skills Assessment

Luke R. Teran

Contents

[1. Coding: 3](#_Toc165378659)

[a. Write a function that takes input of two numbers 𝑎 and 𝑏 . 3](#_Toc165378660)

[b. Give the code for generating 100 random values sampled from a Gaussian 3](#_Toc165378661)

[c. Write a function to find the maximum and minimum element of an unsorted 3](#_Toc165378662)

[2. Data Analytics: 8](#_Toc165378663)

[a. Calculate the signed errors between the mfg spec and manufactured 8](#_Toc165378664)

[b. Assuming the mfg spec is the desired length, calculate the unsigned 8](#_Toc165378665)

[c. Calculate the standard deviation, σ, of the signed percent error 8](#_Toc165378666)

[d. What percent of the values fall within 1σ, 2σ, and 3σ bounds 8](#_Toc165378667)

[3. Data Structures 11](#_Toc165378668)

[What are some data structures with which you are familiar? Discuss their adv. 11](#_Toc165378669)

[4. Machine Learning (Answer any 3 of the following): 12](#_Toc165378670)

[a. Why is reproducibility an important concept in machine learning software dev? 12](#_Toc165378671)

[b. What methods can be used to identify outliers within a dataset? 12](#_Toc165378672)

[c. How do you test whether the data meets the assumptions of linear regression? 12](#_Toc165378673)

[5. Binary conversion and memory storage: 13](#_Toc165378674)

[a. What is the minimal number of bits needed to represent the decimal 1898 in 13](#_Toc165378675)

[b. What is the largest value that can be stored in an unsigned int? 13](#_Toc165378676)

[c. In what cases would you use a double instead of a float? 13](#_Toc165378677)

[d. Convert 0.1 to binary. What’s a problem with storing certain decimal values in 13](#_Toc165378678)

[e. There is a town in Texas where all families have 4 children. Starting with 8 couples. 14](#_Toc165378679)

[6. Data Analytics, Experiment Design, and Model Building (Open Short Answer): 15](#_Toc165378680)

[Initial thoughts 16](#_Toc165378681)

[Challenges 16](#_Toc165378682)

[Solutions 16](#_Toc165378683)

[Testing System Effectiveness 16](#_Toc165378684)

[In-Depth Approach: 17](#_Toc165378685)

[7. Image Processing: 18](#_Toc165378686)

[a. What are some of the differences in need and application between machine 18](#_Toc165378687)

[b. You take a picture using a high‐end camera. The raw data is captured 18](#_Toc165378688)

[c. What are practical challenges one must consider when working with JPG vs PNG 18](#_Toc165378689)

[d. Looking at the photo of a Triumph TR6 below: 19](#_Toc165378690)

[1.2 Math 20](#_Toc165378691)

[a. Given the system and real valued 𝑥, 𝑦 : 20](#_Toc165378692)

[b. You have 10 unique lines lying in a 2-D plane which are described by 20](#_Toc165378693)

[c. What method can help you find the optimal solution to the system? 20](#_Toc165378694)

[d. Imagine you have N unique planes in 3-D space represented by a linear 20](#_Toc165378695)

[e. For matrix 𝐴, vector 𝐯, and scalar 𝜆: 𝐴𝐯 = 𝜆𝐯. What is 𝐯 with respect to 𝐴? 20](#_Toc165378696)

[2. Normalization and Unit Vectors: 21](#_Toc165378697)

[3. Probability: 21](#_Toc165378698)

[4. Trigonometry: 22](#_Toc165378699)

[References 23](#_References)

## 1. Coding:

### a. Write a function that takes input of two numbers 𝑎 and 𝑏 and returns-1 if 𝑎 < 𝑏, 0 if 𝑎 ==𝑏 and +1 if 𝑎> 𝑏. Test it with a few inputs and provide the output.

### b. Give the code for generating 100 random values sampled from a Gaussian distribution with  a mean of 3.5 and a standard deviation of 0.7. Print and verify the sample mean and  standard deviation of your set approach the expected values. You don’t need to print out  the list of random values.

### c. Write a function to find the maximum and minimum element of an unsorted integer array. Test the function by generating a random array of 16 elements. Provide the output.

**Answer**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedOutput 1

Output 2A screenshot of a computer

Description automatically generated

Output 3

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* File:        header.h

 \* Author:      Luke Teran

 \* Date:        24 April 2024

 \* Description: Includes necessary headers, defines global constants and macros,

 \*prototypes functions, and declares ArrayData class

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Include necessary headers here

#ifndef HEADER\_H

#define HEADER\_H

#include <algorithm>

#include <cstdlib>

#include <iostream>

#include <random>

#include <vector>

using namespace std;

// Define any global constants or macros here

const int SAMPLE\_SIZE = 100;

const int RANDOM\_ARRAY\_SIZE = 16;

// Define any global functions or classes here

signed int fn\_a(int a, int b);

void fn\_b(float mean, float std\_dev, const int \*);

float std\_dev\_calc(float sample\_mean, float sample[], const int \*SAMPLE\_SIZE);

void console\_border();

// Implementation of functions or classes, if applicable

class ArrayData {

private:

  std::vector<int> data;

  int min\_value, max\_value;

public:

  void random\_array\_generator(const int &RANDOM\_ARRAY\_SIZE);

  void MinMaxUpdate();

  void printValues();

  void printMinMax();

};

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* File:        main.cpp

 \* Author:      Luke Teran

 \* Date:        24 April 2024

 \* Description: Integrates header.h, functions.cpp for desired output

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Include necessary headers here

#include "header.h"

// Define any global constants or macros here

extern const int RANDOM\_ARRAY\_SIZE;

// Implementation of functions or classes, if applicable

int main() {

  cout << "Author: luketeran@tamu.edu" << endl;

  // fn\_a

  console\_border();

  int a, b;

  cout << "input a: ";

  cin >> a;

  cout << "input b: ";

  cin >> b;

  cout << "function a output:"

       << "\t" << fn\_a(a, b) << endl;

  // fn\_b

  console\_border();

  const float MEAN = 3.5, STD\_DEV = 0.7;

  fn\_b(MEAN, STD\_DEV, &SAMPLE\_SIZE);

  // fn\_c

  console\_border();

  ArrayData randomArray;

  randomArray.random\_array\_generator(RANDOM\_ARRAY\_SIZE);

  randomArray.MinMaxUpdate();

  randomArray.printValues();

  randomArray.printMinMax();

  return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* File:        functions.cpp

 \* Author:      Luke Teran

 \* Date:        24 April 2024

 \* Description: Implementation of header.h fn prototypes. Written to meet

 \* requirements of "Candidate Technical Assessment 2024.pdf".

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Include necessary headers here

#include "header.h"

// Define any global constants or macros here

extern const int SAMPLE\_SIZE;

// Implementation of functions or classes, if applicable

signed int fn\_a(int a, int b) {

  int c = a - b;

  if (c > 0) {

    return 1;

  } else if (c < 0) {

    return -1;

  }

  return 0;

}

void fn\_b(float mean, float STD\_DEV, const int \*) {

  float sample\_mean = 0, sample\_std\_dev, sample[SAMPLE\_SIZE];

  std::random\_device rd{};

  std::mt19937 gen{rd()};

  std::normal\_distribution<float> d(mean, STD\_DEV);

  for (int sample\_index = 0; sample\_index < SAMPLE\_SIZE; sample\_index++) {

    sample[sample\_index] = d(gen);

    // cout << "[" << sample\_index+1 << "]: " << sample[sample\_index] << endl;

    sample\_mean += sample[sample\_index];

  }

  sample\_mean /= SAMPLE\_SIZE;

  sample\_std\_dev = std\_dev\_calc(sample\_mean, sample, &SAMPLE\_SIZE);

  cout << "Sample Mean:"

       << "\t"

       << "\t" << sample\_mean << std::endl

       << "Standard Deviation:"

       << "\t" << sample\_std\_dev << std::endl;

}

float std\_dev\_calc(float sample\_mean, float sample[], const int \*SAMPLE\_SIZE) {

  float numerator = 0, std\_dev;

  for (int sample\_index = 0; sample\_index < \*SAMPLE\_SIZE; sample\_index++) {

    numerator += pow((sample[sample\_index] - sample\_mean), 2);

  }

  std\_dev = sqrt(numerator / (\*SAMPLE\_SIZE - 1));

  return std\_dev;

}

void console\_border() {

  for (int border = 0; border < 20; border++) {

    cout << "-";

  }

  cout << endl;

}

void ArrayData::random\_array\_generator(const int &RANDOM\_ARRAY\_SIZE) {

  data.clear();

  srand(time(NULL));

  std::vector<int> random\_array;

  for (int random\_array\_index = 0; random\_array\_index < RANDOM\_ARRAY\_SIZE;

       random\_array\_index++) {

    random\_array.push\_back(rand());

  }

  data = random\_array;

}

void ArrayData::MinMaxUpdate() {

  min\_value = \*std::min\_element(data.begin(), data.end());

  max\_value = \*std::max\_element(data.begin(), data.end());

}

void ArrayData::printValues() {

  for (int array\_index = 0; array\_index < data.size(); array\_index++) {

    cout << "[" << array\_index + 1 << "]:"

         << "\t" << data.at(array\_index) << endl;

  }

};

void ArrayData::printMinMax() {

  cout << "Maximum element:"

       << "\t" << max\_value << endl;

  cout << "Minimum element:"

       << "\t" << min\_value << endl;

};

## 2. Data Analytics:

### a. Calculate the signed errors between the mfg spec and manufactured part lengths. What is the average error?

**Answer**

*Excel*

|  |
| --- |
| Average Error |
| 0.230769231 |

=AVERAGE(Mfg\_length\_data1!D:D)

### b. Assuming the mfg spec is the desired length, calculate the unsigned percent error and sort in ascending order. Print the Part ID values associated with the 10 largest percent errors.

**Answer**

*Excel*

|  |
| --- |
| 10 Largest Percent Errors |
| A33 |
| A32 |
| A39 |
| A17 |
| A31 |
| A23 |
| A35 |
| A15 |
| A05 |
| A07 |

=XLOOKUP(  
LARGE(  
Mfg\_length\_data!$G:$G,  
COUNTA(Mfg\_length\_data!$A$1:A1)),  
  
Mfg\_length\_data!$G:$G,Mfg\_length\_data!$A:$A,"FAIL",0,1)

### c. Calculate the standard deviation, σ, of the signed percent error

**Answer**

*Excel*

|  |
| --- |
| Signed Percent Error Std Dev |
| 5.745486437 |

=STDEV.P(Mfg\_length\_data!E:E)

### d. What percent of the values fall within 1σ, 2σ, and 3σ bounds (looking at percent error)

**Answer**

*Excel*

|  |  |  |
| --- | --- | --- |
| 1 sigma | 2 sigma | 3 sigma |
| 59% | 100% | 100% |

**SQL Query:**

SELECT

Mfg\_length\_data.[Part ID],

Mfg\_length\_data.[Length-Mfg Spec-mm],

Mfg\_length\_data.[Length-Actual-mm],

[Length-Actual-mm]-[Length-Mfg Spec-mm] AS [Signed Error], ([Length-Actual-mm]-[Length-Mfg Spec-mm])/[Length-Mfg Spec-mm]\*100 AS [Percent Signed Error],

Abs([Length-Actual-mm]-[Length-Mfg Spec-mm]) AS [Unsigned Error],

Abs(([Length-Actual-mm]-[Length-Mfg Spec-mm])/[Length-Mfg Spec-mm])\*100 AS [Unsigned Percent Error]

FROM

Mfg\_length\_data

GROUP BY

Mfg\_length\_data.[Part ID],

Mfg\_length\_data.[Length-Mfg Spec-mm],

Mfg\_length\_data.[Length-Actual-mm];

**Microsoft Access - Mfg\_length\_data worksheet (linked to mfg\_length\_data.accdb)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Part ID** | **Length-Mfg Spec-mm** | **Length-Actual-mm** | **Signed Error** | **Percent Signed Error** | **Unsigned Error** | **Unsigned Percent Error** |
| A04 | 117 | 117 | 0 | 0 | 0 | 0 |
| A22 | 96 | 96 | 0 | 0 | 0 | 0 |
| A27 | 92 | 92 | 0 | 0 | 0 | 0 |
| A18 | 145 | 146 | 1 | 0.689655172 | 1 | 0.689655172 |
| A12 | 112 | 111 | -1 | -0.892857143 | 1 | 0.892857143 |
| A03 | 95 | 96 | 1 | 1.052631579 | 1 | 1.052631579 |
| A20 | 86 | 87 | 1 | 1.162790698 | 1 | 1.162790698 |
| A19 | 83 | 82 | -1 | -1.204819277 | 1 | 1.204819277 |
| A38 | 146 | 144 | -2 | -1.369863014 | 2 | 1.369863014 |
| A09 | 61 | 60 | -1 | -1.639344262 | 1 | 1.639344262 |
| A13 | 61 | 60 | -1 | -1.639344262 | 1 | 1.639344262 |
| A28 | 57 | 56 | -1 | -1.754385965 | 1 | 1.754385965 |
| A30 | 56 | 57 | 1 | 1.785714286 | 1 | 1.785714286 |
| A02 | 98 | 100 | 2 | 2.040816327 | 2 | 2.040816327 |
| A26 | 121 | 124 | 3 | 2.479338843 | 3 | 2.479338843 |
| A21 | 73 | 75 | 2 | 2.739726027 | 2 | 2.739726027 |
| A34 | 59 | 61 | 2 | 3.389830508 | 2 | 3.389830508 |
| A06 | 81 | 78 | -3 | -3.703703704 | 3 | 3.703703704 |
| A10 | 71 | 68 | -3 | -4.225352113 | 3 | 4.225352113 |
| A29 | 144 | 151 | 7 | 4.861111111 | 7 | 4.861111111 |
| A37 | 81 | 77 | -4 | -4.938271605 | 4 | 4.938271605 |
| A36 | 90 | 95 | 5 | 5.555555556 | 5 | 5.555555556 |
| A08 | 142 | 134 | -8 | -5.633802817 | 8 | 5.633802817 |
| A14 | 88 | 83 | -5 | -5.681818182 | 5 | 5.681818182 |
| A16 | 70 | 74 | 4 | 5.714285714 | 4 | 5.714285714 |
| A25 | 90 | 96 | 6 | 6.666666667 | 6 | 6.666666667 |
| A24 | 81 | 87 | 6 | 7.407407407 | 6 | 7.407407407 |
| A11 | 53 | 57 | 4 | 7.547169811 | 4 | 7.547169811 |
| A01 | 126 | 136 | 10 | 7.936507937 | 10 | 7.936507937 |
| A07 | 108 | 117 | 9 | 8.333333333 | 9 | 8.333333333 |
| A05 | 71 | 77 | 6 | 8.450704225 | 6 | 8.450704225 |
| A15 | 69 | 75 | 6 | 8.695652174 | 6 | 8.695652174 |
| A35 | 149 | 136 | -13 | -8.724832215 | 13 | 8.724832215 |
| A23 | 122 | 111 | -11 | -9.016393443 | 11 | 9.016393443 |
| A31 | 54 | 49 | -5 | -9.259259259 | 5 | 9.259259259 |
| A17 | 107 | 117 | 10 | 9.345794393 | 10 | 9.345794393 |
| A39 | 128 | 116 | -12 | -9.375 | 12 | 9.375 |
| A32 | 72 | 79 | 7 | 9.722222222 | 7 | 9.722222222 |
| A33 | 129 | 116 | -13 | -10.07751938 | 13 | 10.07751938 |

## 3. Data Structures

### What are some data structures with which you are familiar? Discuss their advantages and disadvantages and example applications where they are best suited.

*Arrays, Vectors (Dynamic Arrays), Matrices, Linked Lists, Stacks & Queues, Hash Maps*

**Arrays** allow for resource utilization to be minimized for environments where resources are extremely limited, but the management of memory allocation and deallocation is complex, allowing for memory leaks if any bugs exist. Best used in low-cost embedded systems.

**Vectors** or built in Dynamic Arrays in some languages allow for automatic resizing of arrays and provides member functions to access information about the array to speed up the software development process, but vectors are not as suitable for environments where resources are extremely limited. Best used in place of arrays in modern computers.

**Matrices** are two-dimensional arrays which can be represented using libraries like the Eigen library in C++ which provides specialized classes with added functionality and operations. Matrices also provide a framework for linear algebra, numerical operations and parallelization. Efficient parallelization is especially important for faster computation of large matrix-based algorithms or tasks when dealing with extensive amounts of data. Disadvantages could include memory overhead issues like memory fragmentation in memory-constrained environments and computational complexity.

**Linked Lists** are advantageous with their ability to be resized dynamically with a linear time complexity O(n), efficient insertion and deletions with linear or constant time complexity and memory efficiency. Disadvantages include that the linked list must be traversed sequentially to find the intended target element, more memory is used to store the pointers that link nodes, and increased space complexity.

**Stacks and Queues** offer LIFO or FIFO collections of elements which allow for sequential operations but lack the ability to access elements in the middle of the stack or queue. Queues also help with concurrent programming and multi-threading.

**Hash Maps** allow for constant-time average-case complexity for insertion, deletion, and lookup operations, along with dynamic sizing, but they have disadvantages such as hash collisions, memory overhead requirements, unordered iteration and dependence on the hash function.

## 4. Machine Learning (Answer any 3 of the following):

### a. Why is reproducibility an important concept in machine learning software development?

**Answer**

Reproducibility is necessary to confirm research findings, reduce risk of errors, increase reliability of experiments. It allows independent verification and validation of results. [6]

### b. What methods can be used to identify outliers within a dataset?

**Answer**

- Visually: histogram, scatter plot

- Interquartile Range: Outliers are 1.5 \* IQR above Q3 or 1.5 \* IQR below Q1

- Unsupervised Cluster-Based Detection Method [1]

### c. How do you test whether the data meets the assumptions of linear regression?

**Answer**

1. Look at Residual vs Fitted Plot. The Lowess Smoother line should be randomly distributed around the horizontal line at zero. [4], [5]
2. Use Durbin-Watson test to check if predictors are independent & observed with negligible error. [4, 5]
3. Look at Residual vs Fitted Plot. The Lowess Smoother line should be flat on 0, indicating residual error have a mean value of zero. [4], [5]
4. Check data for Homoscedasticity. Scale-Location Plot: Residual points should be equally spread around the line. Breusch-Pagan test or White test. [4], [5]

## 5. Binary conversion and memory storage:

### a. What is the minimal number of bits needed to represent the decimal 1898 in binary?

**Answer**

0111 0110 1010 = 1898

11 bits can represent the decimal.

### b. What is the largest value that can be stored in an unsigned int?

**Answer**

Unsigned int max = 2^BitSize – 1

UINT32\_MAX = 2^32 – 1 = 4,294,967,295

UINT16\_MAX = 2^16 – 1 = 65,535

### c. In what cases would you use a double instead of a float?

**Answer**

When precision is needed or when a wider range of values (larger or smaller than 32-bit floats can provide) is needed, and when memory is not a major concern (double uses twice as much memory as float).

### d. Convert 0.1 to binary. What’s a problem with storing certain decimal values in binary?

**Answer**

Certain decimals cannot be represented precisely in binary. As shown below, decimal 0.1 repeats binary decimal places infinitely.

0.1 \* 2

0.2 -> 0.0

0.2 \* 2

0.4 -> 0.00

0.4 \* 2

0.8 -> 0.000

0.8 \* 2

1.6 -> 0.0001

0.6 \* 2

1.2 -> 0.00011

0.2 \* 2

0.4 -> 0.000110

0.4 \* 2

0.8 -> 0.0001100

0.8 \* 2

1.6 -> 0.00011001

0.6 \* 2

1.2 -> 0.000110011

0.2 -> 0.000110011

0.2 \* 2

0.4 -> 0.0001100110

0.4 \* 2

0.8 -> 0.00011001100

…

### e. There is a town in Texas where all families have 4 children. Starting with 8 couples, and assuming monogamous couples which mate for life, what data type and/or how many bits of memory are necessary to store the number of children at the end of 12 generations.

**Answer**

Data Type: Unsigned Long Integer (32-bit unsigned integer)

Minimum bits necessary to store decimal 65536 = 17-bits

A screenshot of a table

Description automatically generatedA table of maths

Description automatically generated

Number of 12th gen kids = 65536

65536 = 0001 0000 0000 0000 0000

Unsigned short int max = 2^16 – 1 = 65,535

Unsigned long int max = 2^32 – 1= 4,294,967,295

## 6. Data Analytics, Experiment Design, and Model Building (Open Short Answer):

***Prompt***: You're designing a system which processes a wide variety of video data and derives contextual labels and descriptions of the scene. Every input video is accompanied by meta‐data parameters like time of day, camera optics info, geographic location, and others.

Your system has many tuning parameters which define its configuration. Some configurations work better for certain sub‐sets (classes) of the data space, so an ideal system would be smart and adapt based on the input video and accompanying meta‐data. However, neither the classes nor their optimal system configurations are known *apriori*. Due to the many parameters and the range of values they span, the configuration space is too massive to be explored exhaustively. Also, some tuning parameters are more impactful than others, but you don’t necessarily know which ones.

**Question**: How do you approach this problem to design the best performing system? Describe some of the challenges and how you can overcome them. How do you test your system is working?

**Possible Useful Terms**: clustering, optimal, tractability, sensitivity, feasibility, control variable, fix, baseline, experiment, exhaustive, brute‐force search, heuristic, dimensionality, space, scope, PCA, basis, sub‐sampling, classes, combinatorial complexity, correlation, local, global, loss function, residual.

### Initial thoughts

Based on the many unknowns and inability for supervisors to know which tuning parameters are most impactful than others, this problem should be approached with either unsupervised or reinforcement learning. Configurations should be organized in some order such as ascending in order of one parameter’s value. The machine learning process should be focused on grouping data into clusters and once a cluster is identified, then resources should be spent on exploring differing configurations with a meaningful variation in at least one parameter until supervisor can identify a positive “good” output. Once a supervisor identifies an output as positive, the machine should explore other configurations with a value similar to the parameter’s value in the “good” configuration. Outliers, or inputs which do not belong to a cluster or belong to a fringe cluster which has only a few data points should be handled last and separately. This method strongly follows a positive reinforcement learning process while taking advantage of clustering methods.

### Challenges

1. Complex Configuration Space
2. Unknown Optimal Configurations
3. Data Heterogeneity
4. Parameters of Unknown Significance

### Solutions

1. Sampling and Subsetting
2. Adaptive Algorithms
3. Clustering and Classification
4. Sensitivity Analysis

### Testing System Effectiveness

1. Cross-Validation:
   1. Division of dataset into training and testing sets to evaluate the system’s performance on unseen data.
2. Continuous Monitoring
   1. Implementation of system for monitoring key performance indicators (KPIs) and setting up alerts for deviations from expected behavior.

### In-Depth Approach:

Initially PCA should be used to reduce the dimensionality of the data (metadata).

Clustering algorithms can be used to identify patterns in data and group similar videos together to create sub-sets of data. Within these sub-sets, sub-sampling can be done to explore the configuration space.

An adaptive system should be developed to dynamically adjust based on the input video and metadata using heuristics to guide selection of configuration parameters. Reinforcement learning techniques could be used to learn optimal configurations over time.

Experiments to evaluate performance of different configurations on various subsets of data should be conducted. This involved defining evaluation metrics and comparing the performance of different configurations using these metrics.

Sensitivity analysis can be performed to identify the tuning parameters that have the most significant impact on the performance.

A baseline configuration can be established to compare against experimental configurations.

With the vast configuration space, it is not feasible to exhaustively explore every possible combination. Instead, strategies like greedy search or evolutionary algorithms for learning should be used to traverse the space and identify “good” or promising configurations.

While designing the system, local and global optimization should be considered in balance. To do this, iterative refinement of configurations should take place based on performance feedback.

[1], [2], [3]

## 7. Image Processing:

### a. What are some of the differences in need and application between machine vision imaging cameras and hobbyist photography cameras?

Machine vision imaging cameras prioritize contrast, sharpness and grayscale accuracy over color reproduction, and they are typically optimized for high-speed imaging (higher frame rates, minimal shutter lag and rapid data transfer) while hobbyist cameras prioritize color fidelity, dynamic range, and resolution. Machine vision cameras also are designed to integrate with industrial automation systems and computer vision software using standard protocols and come with SDKs for integrating with custom applications while hobbyist cameras use consumer-grade communication protocols like Wi-Fi, Bluetooth, or MTP. Machine vision cameras are also purpose-built for their environment to withstand harsh operating conditions while hobbyist cameras are more delicate.

### b. You take a picture using a high‐end camera. The raw data is captured and saved along with a compressed (lossy) copy. The compressed copy has several dark areas of the image which seem to lack contrast and look flat, while there are also some pixels in bright areas of the image which exhibit clipping (saturation). What can you do to correct these problems?

After adjusting the tone curve levels, the raw data can be reprocessed to generate a new image without the same contrast and clipping issues.

### c. What are practical challenges one must consider when working with JPG vs PNG files?

JPG

* Lossy compression
* Smaller file size
* Greater Color depth (24-bit)
* No transparency
* Better for smooth color gradient scenes
* Compression artifacts

PNG does not use lossy compression

* Lossless compression
* Larger file size
* Transparency
* Better for images that require high-quality reproduction such as graphics
* No compression artifacts

### d. Looking at the photo of a Triumph TR6 below:

A collage of a car driving on the road

Description automatically generated

Original Mod1 Mod2

Carefully examine the changes in color and lighting which have been introduced from the Original to Mod1. If you were to manually recreate these effects, what specific filters, effects, and methods would you need? What about going from Mod1 to Mod2?

Original -> Mod1

Exposure or White point increased

Temperature changed to warmer

Contrast increased

Black Vignette large radius

Mod1 -> Mod2

Image broken into 4 quadrants

Q1: Grayscale

Q2: Hue Change

Q3: Selection of leaves which are under certain brightness level->Recolor, Car selection->Glow effect

Q4: Grayscale, Leaf selection->Color inversion, Contrast Decreased and blur of remainder of quadrant

## 1.2 Math

1. Linear Algebra

### Given the system and real valued 𝑥, 𝑦 :

2𝑥 + 3 cos(𝑦) = 7

-𝑥 + 𝑏 cos(𝑦) = 3

solve for 𝑥, 𝑦 under the two cases: 𝑏 = 8, and 𝑏 = 2.

Domain of arccos(x): [-1,1]

arccos(13/7) is imaginary and y must be real. No solution for b = 2.

𝑏 = 8

𝑏 = 2

### You have 10 unique lines lying in a 2-D plane which are described by a system of equations in the form Does the system have a unique solution or is It under/over determined? Is the system matrix invertible? Why, how do you know?

* + 1. The system is under determined. The system matrix is not invertible because it is under determined and does not have a unique solution. A unique solution is required to be invertible.

### What method can help you find the optimal solution to the system?

* + 1. Least squares regression

### Imagine you have N unique planes in 3-D space represented by a linear system. What is N and what are the dimensions/size of the system matrix if it is invertible?

* + 1. N is the number of unique planes.
    2. If invertible, the system matrix is square and dimensions are N-by-N.

### For matrix 𝐴, vector 𝐯, and scalar 𝜆: 𝐴𝐯 = 𝜆𝐯. What is 𝐯 with respect to 𝐴? If there exists a real nonzero 𝜆 which satisfies this equation, what transformation does 𝐴 perform on 𝐯?

* + 1. 𝐯 is the eigenvector of 𝐴.
    2. Scaling transformation.

### Normalization and Unit Vectors:

It’s common practice to store vectors as columns (or rows) in a matrix. You have the following set of  two 3x1 vectors:

Give the magnitudes of each of the vectors as well as a matrix of the normalized unit vectors.

### Probability:

Looking at a string from a random genetic sequence, what is the probability of coming across a codon where at least two nucleotides are the same? You can assume that the sequence is built up of IID random nucleotides. What “paradox” from the study of probability presents a similar problem?

Given: 1 codon has 3 nucleotides where each type of nucleotide has equal probability and 4 types of nucleotide exist.

**Birthday Paradox:** 4 \* P(Codon has at least 2 nucleotides of type A). Higher probability logically makes sense after consideration that there are 4 nucleotide types, increasing the number of relevant events by a scalar of 4.

### Trigonometry:

You have a camera aimed at a fiducial marker 100 meters away. You want to have stereo vision of the scene such that the parallax angle between your cameras is 1 degree when aiming at your fiducial marker. How far away should you place a second camera?

## References

[1] A. Beheshti, “Clustering methods: An introduction to unsupervised learning techniques,” Medium, <https://behesht.medium.com/clustering-methods-101-an-introduction-to-unsupervised-learning-techniques-984020431ae7> (via: <https://archive.is/bFK3E>) (accessed Apr. 30, 2024).

[2] S. Mishra, “Unsupervised Learning and Data Clustering,” Medium, <https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a> (accessed Apr. 30, 2024).

[3] K. Muralidhar, “Outlier detection methods in machine learning,” Medium, <https://towardsdatascience.com/outlier-detection-methods-in-machine-learning-1c8b7cca6cb8> (accessed Apr. 30, 2024).

[4] M. Khan, “A basic guide to testing the assumptions of linear regression in R,” DataDrive, <https://godatadrive.com/blog/basic-guide-to-test-assumptions-of-linear-regression-in-r> (accessed Apr. 29, 2024).

[5] R. Nau, “Testing the assumptions of linear regression,” Statistical forecasting: notes on regression and time series analysis, [https://people.duke.edu/~rnau/testing.htm](https://people.duke.edu/~rnau/testing.htm%20) (accessed Apr. 29, 2024).

[6] Z. Ding and A. Joshi, “Topics in data analysis: 5 – Reproducibility,” Machine Learning Blog | ML@CMU | Carnegie Mellon University, <https://blog.ml.cmu.edu/2020/08/31/5-reproducibility/> (accessed Apr. 29, 2024).